Offline and Online Testability of Random Number Generators

M. Bucci marco.bucci@infineon.com
R. Luzzi raimondo.luzzi@infineon.com

CryptArchi 2019
June 24-25 2019 - Průhonice

P-RNG vs RNG/Entropy-Source Assessment

P-RNG		RNG / Entropy-Source
No Entropy Rate	Feature	Full / Partial Entropy Rate
Make **impossible** Entropy Evaluation	need to	Make **possible** Entropy Evaluation
• Long Memory Sequence • Cryptography	by means of	• Short Memory Sequence • “Short” Hashing
Failure of a battery of Statistical Hypothesis Tests	to be assessed by	Entropy Rate Estimation

Conditional Entropy and Entropy Rate

Assuming L is the memory length of a sequence, the conditional entropy

$$H\left(X_i/X_{i-1}, X_{i-2}, \ldots X_{i-L}\right)$$

is an exact and **complete figure** of the **entropy rate** of the sequence. Namely, **it considers all possible "defects"** (i.e. dependencies).

Indeed, it is derived from the Conditional Probability

$$P\left(X_i/X_{i-1}, X_{i-2}, \ldots X_{i-L}\right)$$

which is, in facts, the state transition table which **completely describe the generator** as a Finite State Machine **whichever it is** (linear, not linear, stochastic or deterministic).

NOTICE: as long as the assumed L is sufficient (i.e. all dependencies are considered), the conditional entropy does not change by splitting, joining or anyhow reversibly transforming the symbols.

Complexity of Conditional Entropy Estimation

Estimating the conditional entropy

$$\hat{H}\left(X_i / X_{i-1}, X_{i-2}, \ldots X_{i-L}\right)$$

implies the joint probability estimation

$$\hat{P}\left(X_i, X_{i-1}, X_{i-2}, \ldots X_{i-L}\right)$$

which results in the complexity

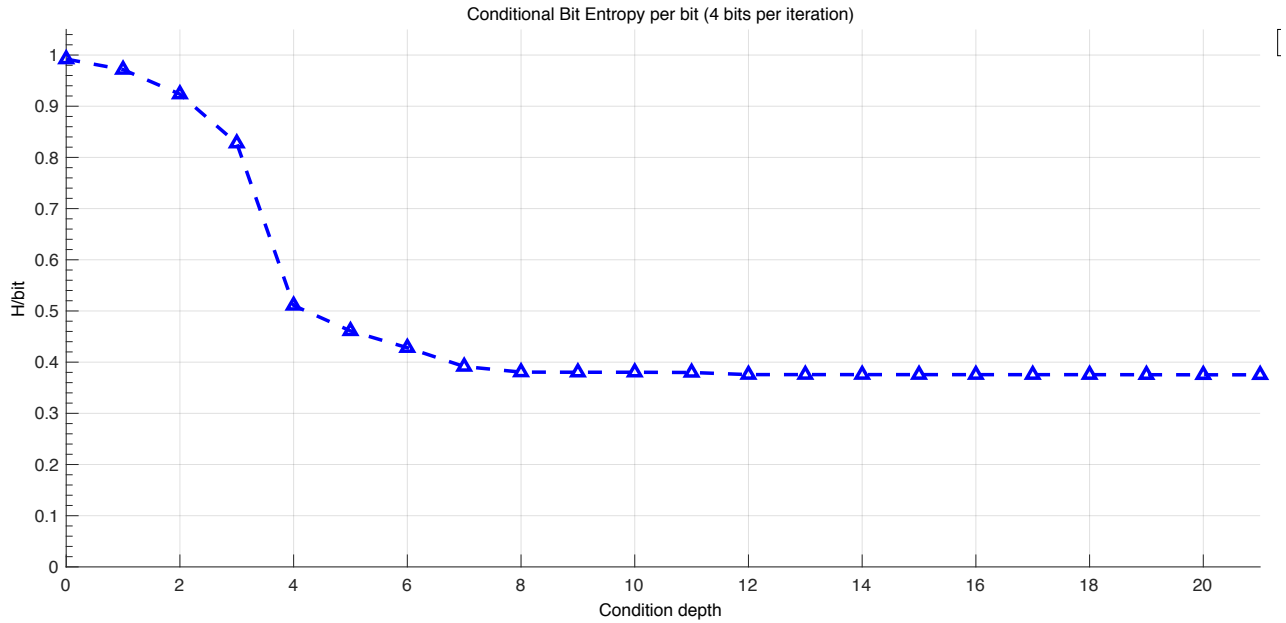
$$O\left(2^{(L+1)\cdot \log_2(\#(X))}\right)$$

i.e. **exponentially growing with memory and symbol bit size**.

This, practically means that:

- **Online tests**: entropy estimation is practically feasible only over mono-bit and short memory sequences.
- **Offline tests**: symbol size and memory length must be anyway small.

Conditional Entropy Estimation: Example (chaotic entropy source)



Conditional entropy estimation of a chaotic entropy source (4 million samples):

- the conditional entropy estimation quickly converges to the correct value
- due to some periodic behaviour, without considering dependencies (condition depth = 0), the sequence would seem to have maximum entropy
- anyway, **the same overestimation would be reported using a Min Entropy estimator**

Conditional probability estimation (contingency table): LFSR example

$P\{x_i=1\}$
0,5

$P\{x_i=1\}$	x_{i-1}
0,5	0
0,5	1

$P\{x_i=1\}$	x_{i-1}	x_{i-2}
0,5	0	0
0,5	1	0
0,5	0	1
0,5	1	1

$P\{x_i=1\}$	x_{i-1}	x_{i-2}	x_{i-3}
0,5	0	0	0
0,5	1	0	0
0,5	0	1	0
0,5	1	1	0
0,5	0	0	1
0,5	1	0	1
0,5	0	1	1
0,5	1	1	1

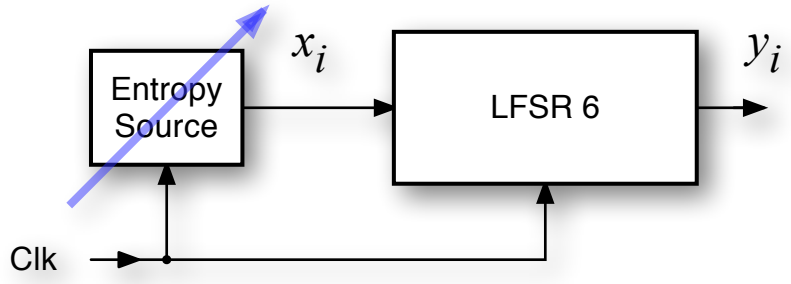
$P\{x_i=1\}$	x_{i-1}	x_{i-2}	x_{i-3}	x_{i-4}
0	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	1	1	0	0
1	0	0	1	0
1	1	0	1	0
1	0	1	1	0
1	1	1	1	0
1	0	0	0	1
1	1	0	0	1
1	0	1	0	1
1	1	1	0	1
0	0	0	1	1
0	1	0	1	1
0	0	1	1	1
0	1	1	1	1

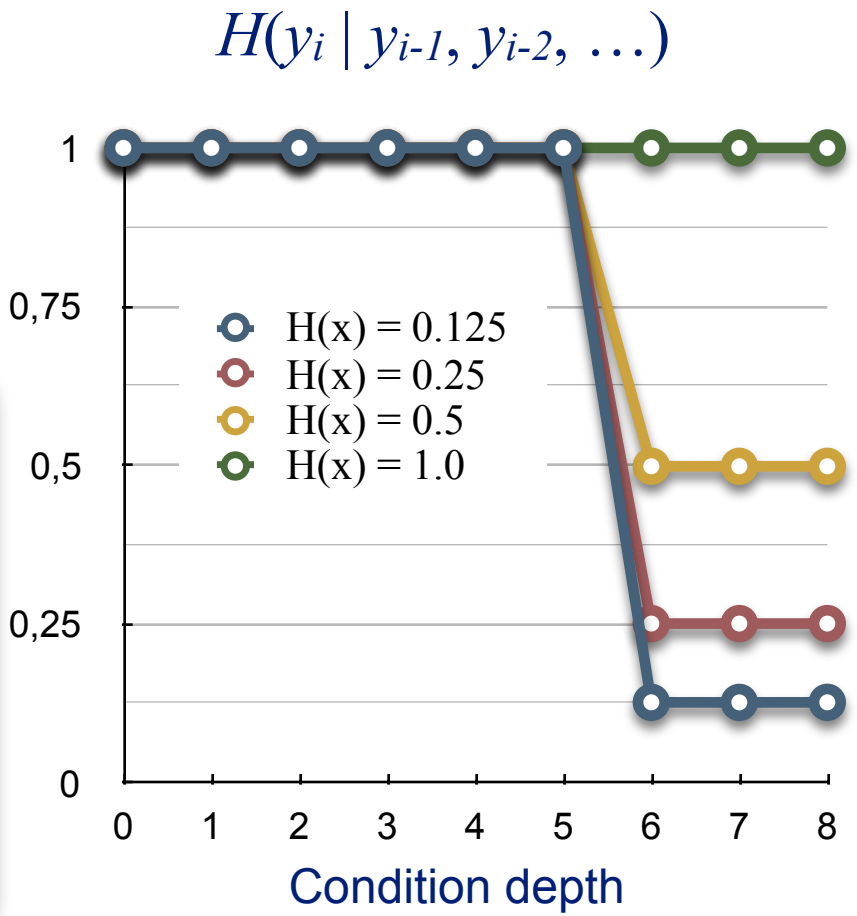
example with LFSR: x^4+x^3+1

- Looks random till depth 3, but it is actually totally deterministic.
- If the system is stochastic, the contingency table never converge to a deterministic value.

The contingency table **completely describes the sequence** (regardless it is stochastic or deterministic), as long as its depth is larger than the sequence memory.

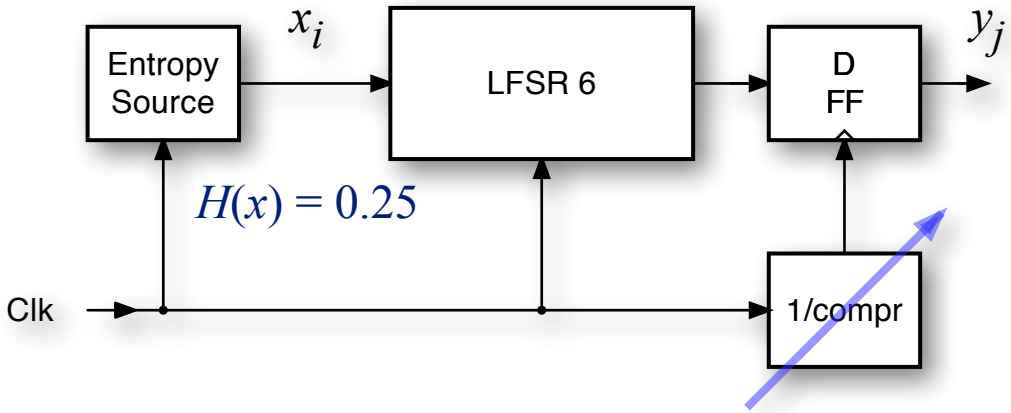
Conditional Entropy estimation after a "short memory" scrambling (reversible transformation)



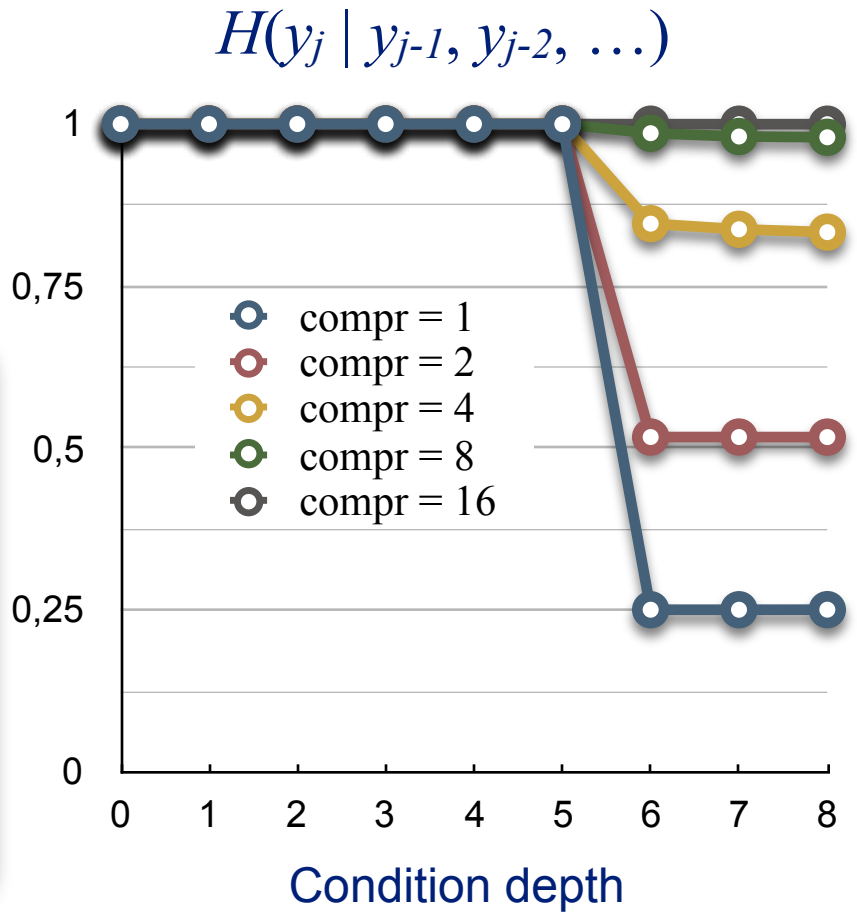


- Output entropy is testable as long as the "depth" of the conditional entropy test is larger than the memory of the scrambler.
- Since the transformation is reversible, the entropy estimated after scrambling is the same as the entropy that enters the scrambler.

Conditional Entropy estimation after a "short memory" compression (non–reversible transformation)



- Output entropy is still testable: despite of decimation, the memory of the sequence is still the same.
- **Virtually, the system behaves as a system which is not decimated but has a larger source entropy.**



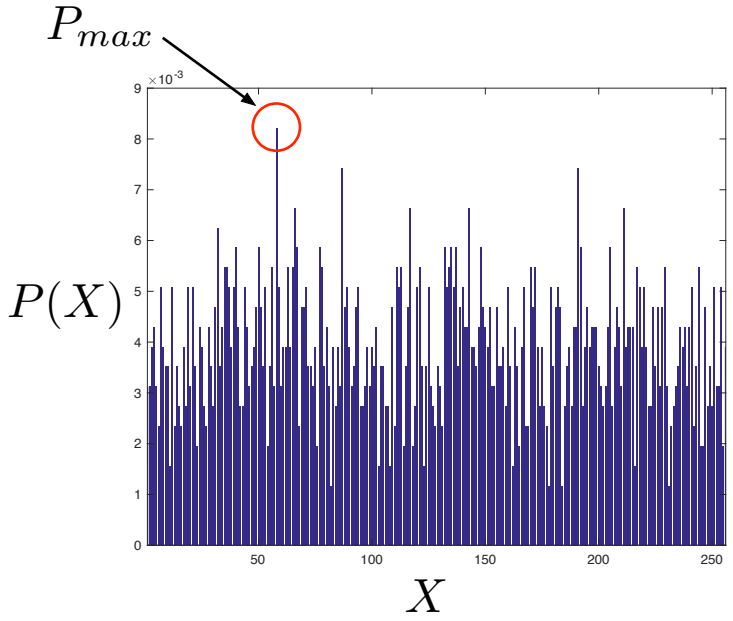
Min Entropy Definition

Min entropy considers just the most likely symbol of a probability distribution:

$$H_{min} = -\log_2 \left(P_{max} \right)$$

Likewise, the min entropy estimator considers just a single (the highest) bin of the empirical histogram

$$\hat{H}_{min} = -\log_2 \left(\hat{P}_{max} \right)$$



Assuming, symbols are independent, it is always: $H_{min} \leq H$

and, because of the estimator variance, it also holds: $\hat{H}_{min} < H_{min}$

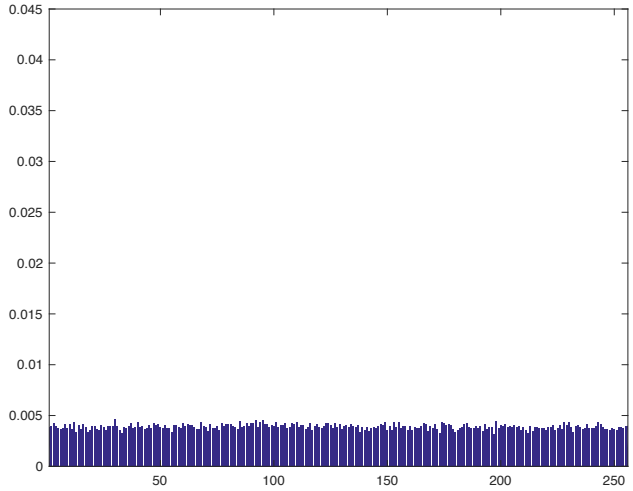
However **also the entropy plugin estimator, is conservative**. it holds: $\hat{H} \leq H$

On the other side, **when symbols are not independent, both estimators can overestimate!**

I.e., in general, **Min Entropy is not conservative** (just a fast, but wrong, estimation).

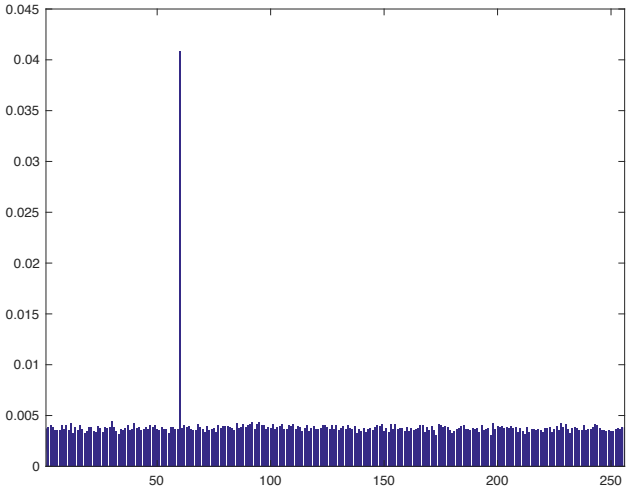
When min entropy can be useful?

Min Entropy can be a quick (just one bin is needed) and sensitive way to detect some “strange” peaks in an empirical distribution.



$$\hat{H} = 7.9968$$

$$\hat{H}_{min} = 7.7473$$

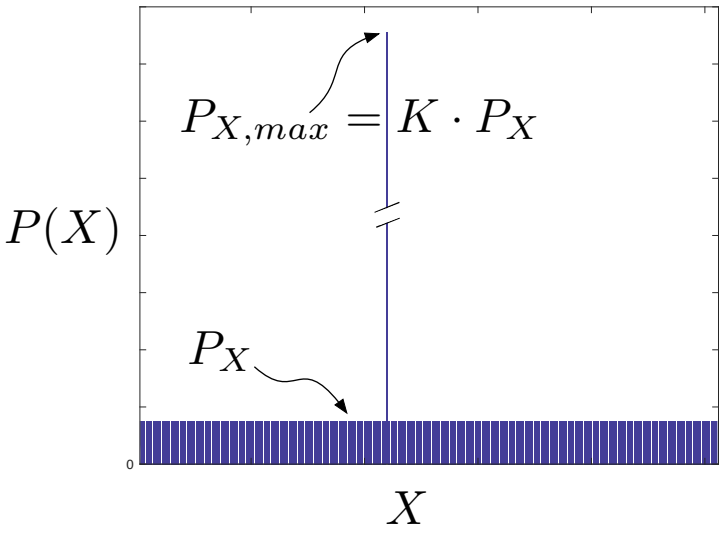


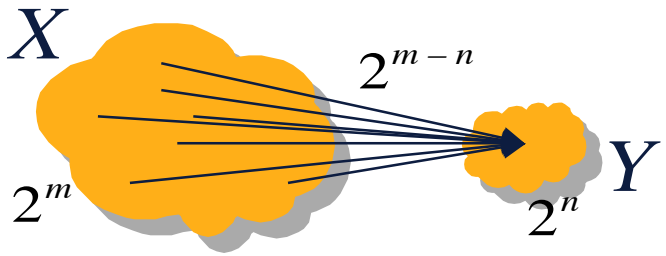
$$\hat{H} = 7.9109$$

$$\hat{H}_{min} = 4.6132$$

However, the “normal” (i.e. the plugin) entropy estimator still underestimate the Shannon entropy.

… but: are probability peaks before hashing post-processing an issue? No!





Assume a *m* ⇒ *n* bit hashing where each output symbol *Y* has the same number *R* = 2^(m-n) of *X* pre-images.

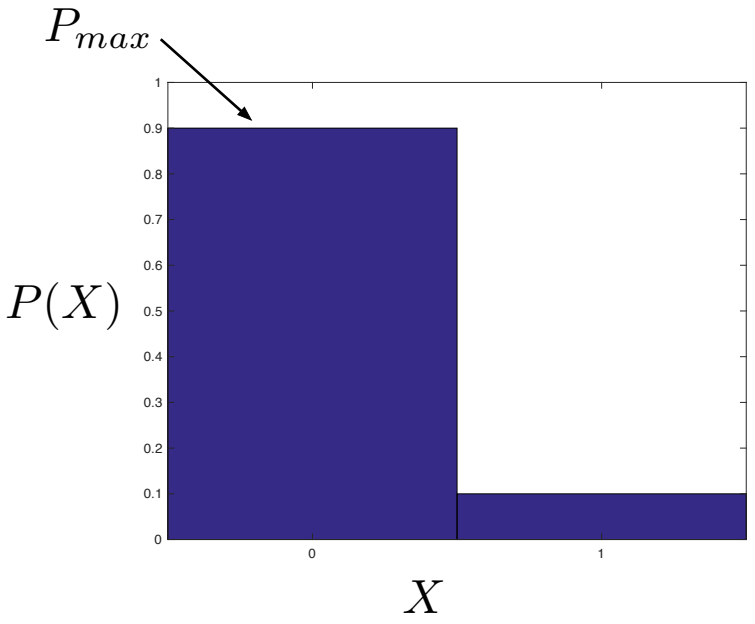
After post-processing hashing: $P_Y = R \cdot P_X;\ P_{Y,max} = (R + K - 1) \cdot P_X;$

$$\Rightarrow \quad \frac{P_{Y,max}}{P_Y} = 1 + \frac{K-1}{R}$$

E.g.: $m = 64;\ n = 8;\ K = 10^4;$ $\Rightarrow \quad \frac{P_{Y,max}}{P_Y} = 1 + 1.38 \cdot 10^{-13}$

OBSERVATION: Min Entropy for binary distributions is a nonsense

In a binary distribution, once P_{max} is defined, the whole distribution, is defined.



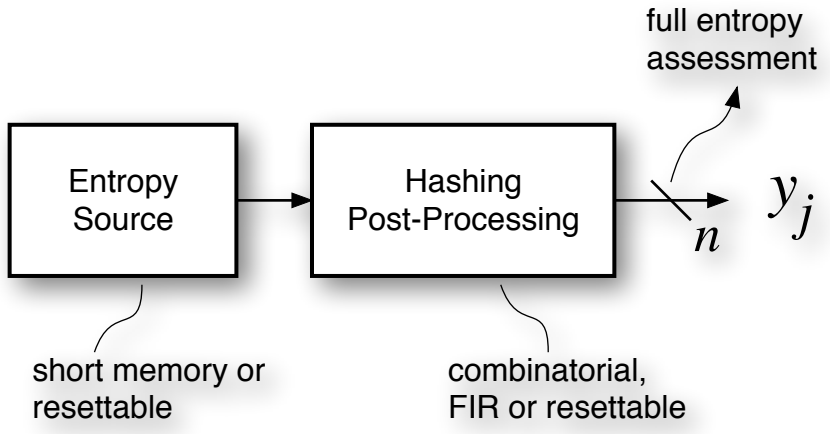
I.e.: $H_{min} \Leftrightarrow H$

Therefore, in this case, Min Entropy is not a conservative estimation, but **just a nonsense** under-estimation.

E.g.: $P\{x = 0\} = 0.9 \quad \Rightarrow \quad \begin{cases} H & = & 0.469 \\ H_{min} & = & 0.152 \end{cases}$

Requirements for Full Entropy Assessment

With a proper design, Full Entropy assessment can be operated directly on the output (i.e. post-processed) data **regardless any evaluation of the source entropy**.



Requirements for **block-based** post-processing (e.g. 8 ≤ n ≤ 16):

because of its large cardinality, Y must be IID, at least in a conservative "**test mode**"

- **source**: resettable
- **post-processing**: combinatorial or resettable

Requirements for **stream-based** post-processing (i.e. n = 1):

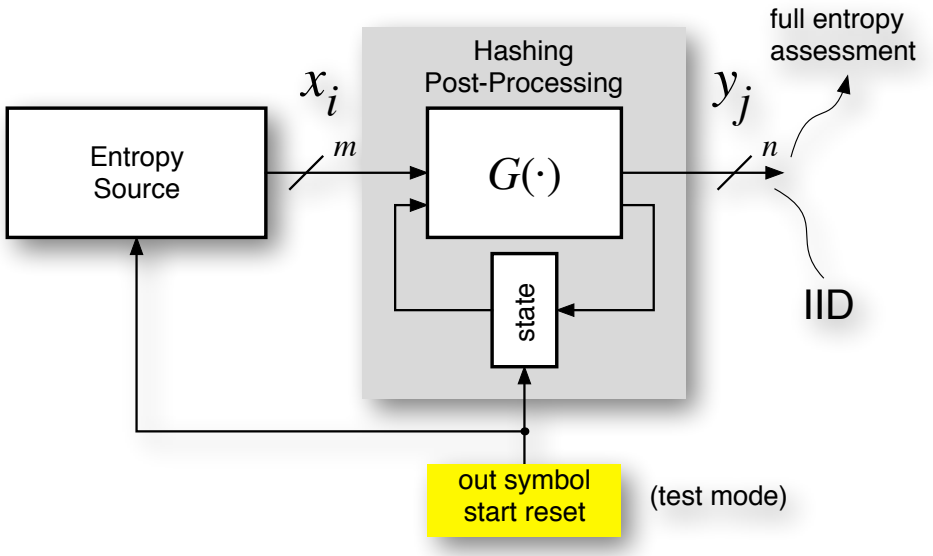
because of its minimal cardinality, Y it is allowed to have some limited memory

- **source**: short memory
- **post processing**: Finite Impulse Response (FIR)

Full Entropy Assessment: Block-Based Post-Processing with Test Mode

Both source and Post-processing are reset before generating each output symbol

Output symbols are IID by construction, regardless any hypothesis on the source except to be **stationary** and **resettable**



State reset can be used only in “test” (i.e. assessment) mode: it can be seen that, in “normal” mode, entropy cannot be less.

Pros:

- this scheme is **applicable to every kind of source and post-processing**

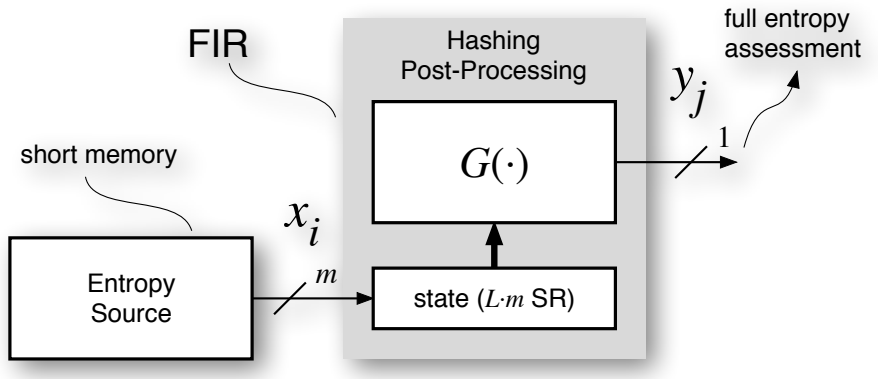
Cons:

- entropy estimation is conservative (entropy is wasted because of resets)
- to be “not too conservative”, m and n must be large enough: e.g. m = 32; n = 8

Full Entropy Assessment: Stream-Based FIR Post−Processing

A short memory source is used with a Finite Impulse Response (FIR) post-processing.

Output symbols have limited memory by construction.



By construction, **a lack of entropy on the input results in a detectable lack of entropy on the output**.

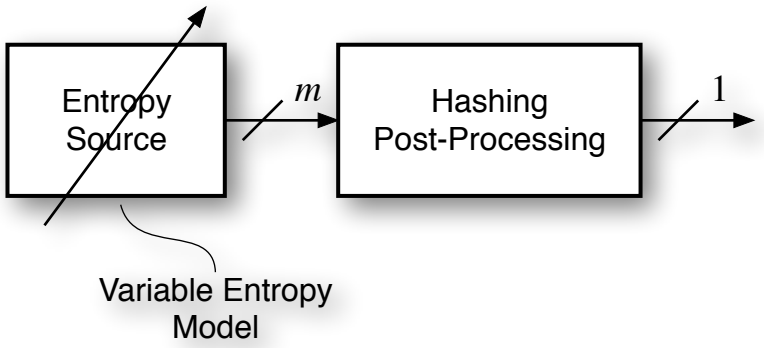
Pros:

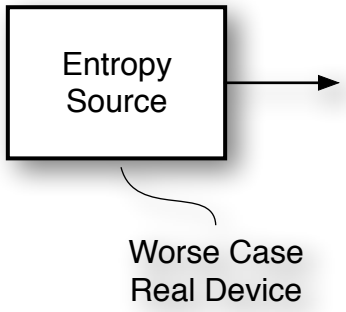
actual output entropy can be assessed directly in operation condition without introducing any conservative test mode

Cons:

a short memory source is required

“Countercheck” of Entropy Assessment with FIR post–processing





Is it really FIR? Which is the minimal entropy required?

A lack of entropy **shall** be detectable when source entropy (or post-processing compression) is reduced (min required source entropy).

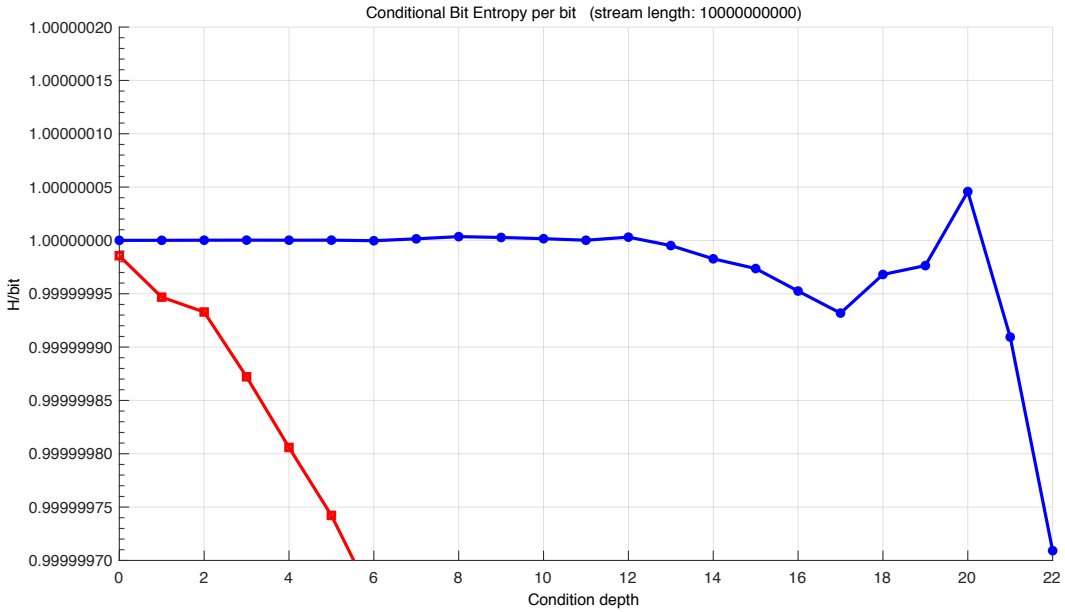
Does the real source provide enough Entropy?

On the real devices, the worse case source entropy **shall** be larger than the min required source entropy.

In order to be conservative with respect of any real experiment, the sizes of the samples used in simulation should be larger than any possible real sample that can be extracted **in stationary conditions** (e.g. 10Gbit).

“Countercheck” of Entropy Assessment with FIR post–processing (example)

Conditional entropy estimation for a chaotic source after FIR post processing (simulated)



★worst case source entropy

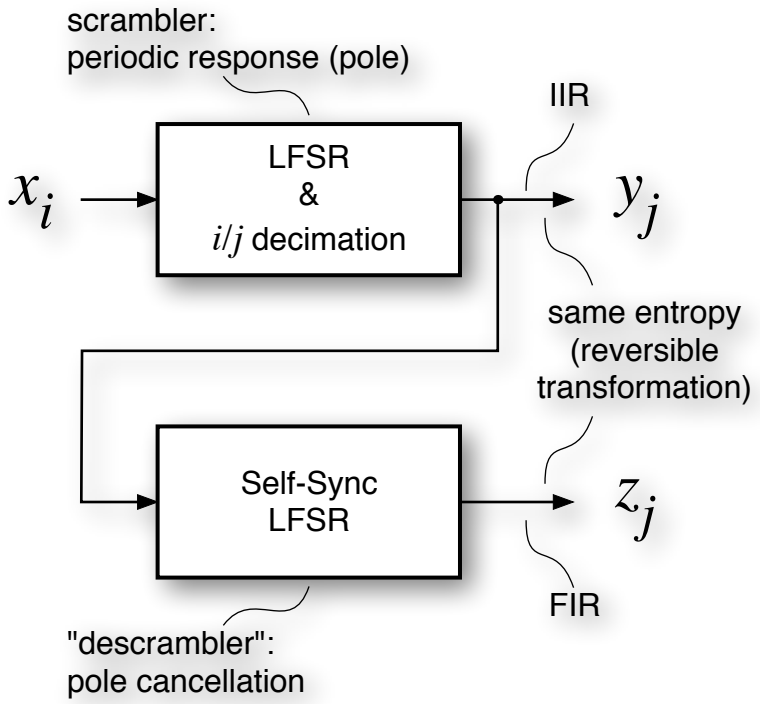
✦ artificially reduced source entropy

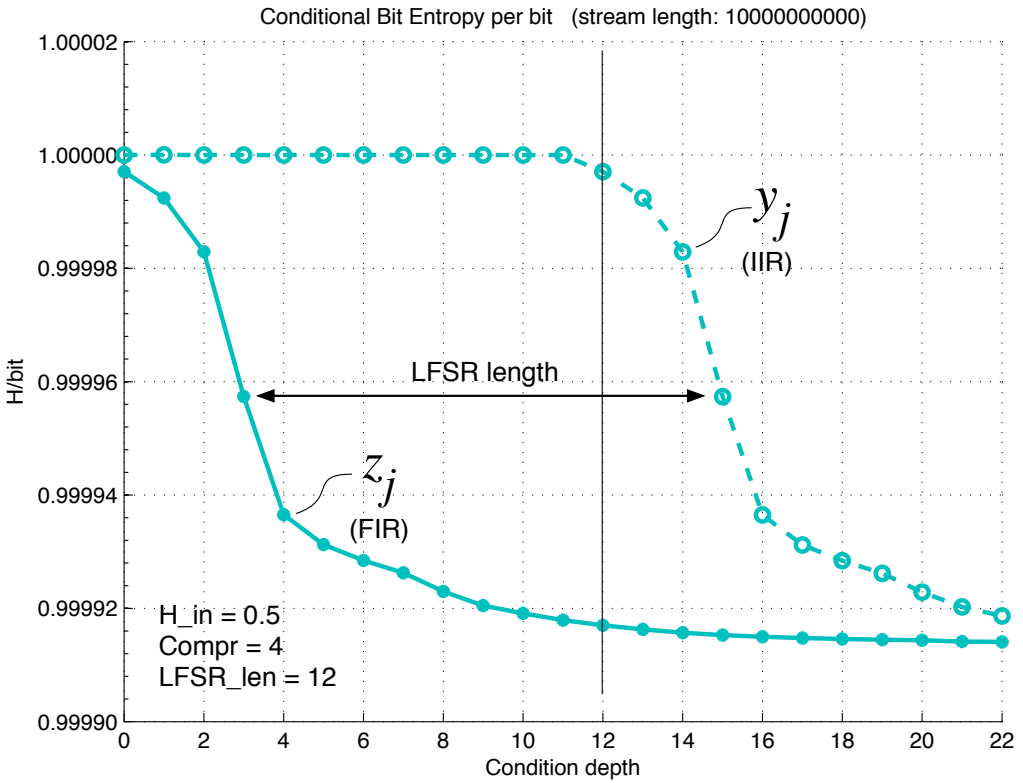
Sample size: 10Gbit

The entropy variation shown by the star marked (blue) line are due just to the variance of the estimator which increases over the condition depth (i.e. test memory length).

On the contrary, the square marked (red) line shows that **a lack of input entropy can be detected by the test despite of the heavy post-processing compression**.

FIR post-processing by cancelling the pole of an IIR "scrambler" (12 bit LFSR example)

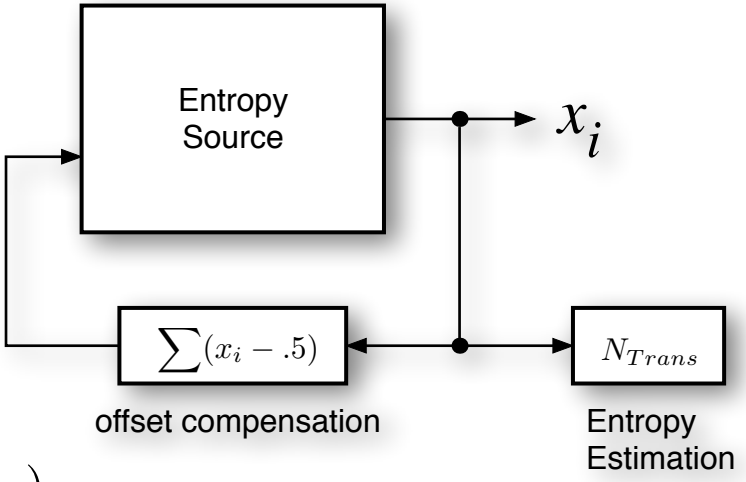




The descrambler has a zero where the scrambler has a pole: periodicity is cancelled, global response is FIR. **Output memory is shortened by the length of the LFSR**.

Online test over an offset-compensated bit source

Due to the offset compensation, this source features a constant sequence at startup and, **in case of lack of noise**, "tends" to produce the alternating sequence …010101…



The quantity

$$N_{Trans} = \sum (x_i \oplus x_{i-2}) \cdot (x_i \oplus x_{i-1} \oplus x_{i-3} \oplus x_{i-4})$$

increases whenever the current bit X_i is "unexpected" with respect of this noiseless model.

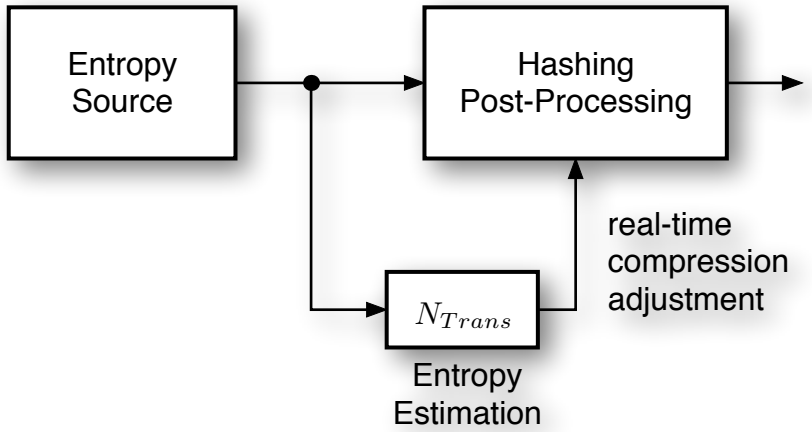
Namely, N_{trans} is the counter of the **discrepancies vs the deterministic model of the source**. Hence, it can be considered as "rough and fast" entropy estimator **for this particular source**.

Notice: the periodic sequence …0101010… which is rejected by this test, will pass both the two health test recommended in NIST 800-90B.

Using online testing for a dynamic adjustment of the post-processing compression

An online entropy estimation can be used to dynamically adjusting the post-processing compression ratio.

In particular, compression can be carried out until a sufficient entropy is collected.



Pros:

- data quality almost constant vs static or dynamic variation of operation conditions and technology parameters;
- false online test alarms are virtually eliminated;
- no data is delivered before being tested;
- a “startup” test is somehow implicit in each data generation: lack of entropy during startup results in a longer, or possibly endless, compression.

Unfortunately, despite of the relevant advantages, this possibility is not considered in NIST 800-90B.

Some good reasons to not use online tests

Beside the question:

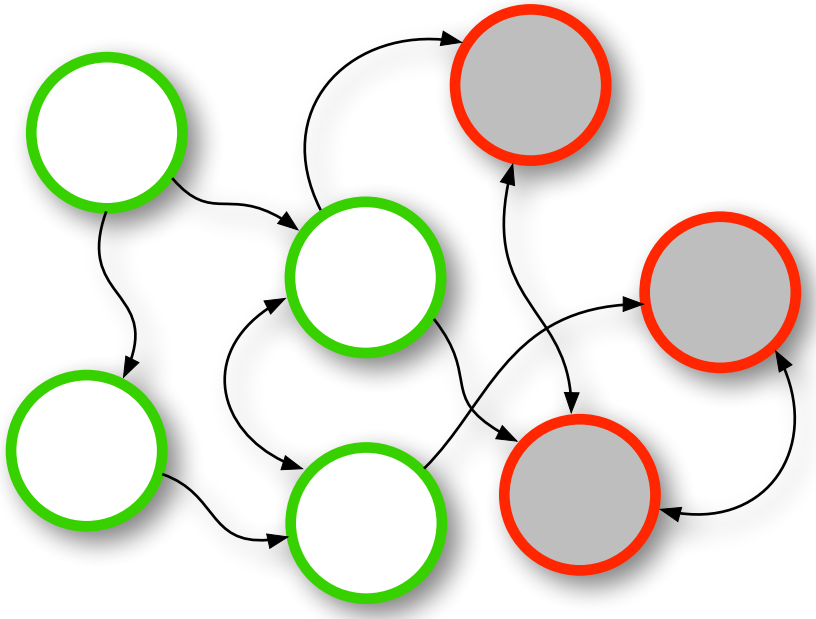
are online tests more robust than what they are supposed to protect (i.e. the entropy sources)?

Some facts:

- Actual coverage against faults/attacks is extremely difficult to be defined
- Ineffective/not-feasible regarding:
  - statistically dependent (e.g. periodic) or large symbols sequences (practically feasible only for mono-bit and very short memory sequences)
  - “fake entropy” (e.g. due to deterministic power supply noise)
  - observation/manipulation attacks (i.e. probing forcing sources)
- Unacceptable delay before detecting anomalies (in the meantime data are delivered)
- False alarms are practically unmanageable: which action should be taken?
- Useful for the attacker: they can help to find sensitive point/issues of the entropy source
- More expensive than redundancy (at the state of the art)

A Reliability-Engineering Approach: designing RNG's as Fault Tolerant Systems

A certain amount of redundancy can allow a system to be reliable/secure in different states where one or more parts are compromised.



Each state can be reached following a certain path of fault-probability/attack-effort.

The set of routes defines the probability/effort required to reach a fault/insecurity state.

NOTICE: a correct design implies an estimation of cost and robustness of the possible components.
E.g.: how much robust and "expensive" are online tests vs redundant entropy sources?
At the state of the art, less robust and more expensive.

Fault/attack robustness by means of redundancy vs online tests

A coverage comparison between comparable cost implementations, shows that source redundancy is much more effective than online tests. E.g.:

Implementation	Reliability Issue			
	single source fault	double fault	single source forcing/observing	false alarm
single–source + online test	✔	✖ (fault on both source and test)	✖	✖
	✖ (low entropy data delivered till alarm assertion)			
2 sources	✔	✖ (fault on both sources)	✔	✔

NOTICE:

- redundancy covers any kind of faults while tests have, typically, a limited coverage;
- even more sources can be used (e.g. 4) with a negligible cost increasing.

Conclusions

- **Full entropy** can be conservatively assessed **directly on the RNG output** (i.e. after hashing post-processing) with **very weak assumptions**:
  1. assuming source and post-processing are **resettable**: with some performance penalty
  2. assuming source and post-processing have **short memory**: without any performance penalty
- **Min Entropy**
  - useless: no advantages vs Shannon entropy (just wrong estimation)
  - meaningless: it is **not conservative**, typically overestimates non IID sequences
  - nonsense in case of binary sequences: it is bijectively related to entropy

- **Online tests**:
  - when they are feasible, they should be used for adaptive post-processing (not for generating, **practically unmanageable**, alarms)
  - in general they are not really effective/usable, **redundant sources offer much better coverage**